DOCUMENT RESUME

ED 290 453 IR 013 109

AUTHOR Nachmias, Rafi; And Others
TITLE A Cognitive-Curricular Model for Teaching Computer Programming to Children. Research Report No. 3.
INSTITUTION Tel-Aviv Univ. (Israel). Computers in Education Research Lab.
PUB DATE Jul 84
NOTE 23p.; For related reports, see IR 013 108-115.
PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Cognitive Processes; Computer Literacy; *Course Descriptions; Curriculum Development; Elementary Secondary Education; Foreign Countries; *Instructional Development; *Microcomputers; *Models; Preadolescents; *Programing; Programing Languages; Psychological Studies; Units of Study
IDENTIFIERS BASIC Programing Language; Cognitive Research; *Israel

ABSTRACT

The hierarchical model for the teaching of computer programming to children that is presented in this report comprises five content domains: (1) technical skills of controlling the machine; (2) mastery of the programming language; (3) understanding the structure of a computer program; (4) understanding the logical dynamics of the computer program; and (5) the knowledge required by the topic of the program. This model was used to construct a teaching unit designed for grades 6 to 9, although in many cases parts of it may be taught as early as grade 4. The teaching unit, which is described in some detail, was used in several situations in 1982-84. The results of these studies are briefly reported with preliminary conclusions on the cognitive, affective, and curricular aspects of the unit. A 13-item bibliography is provided. (EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED290453

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

A Cognitive-Curricular Model for Teaching Computer Programming to Children

Nachmias, Rafi, Mioduser, David, Chen, David

Research Report No. 3
July 1984

The Computers in Education Research Lab.

School of Education

Tel-Aviv University

Tel-Aviv 69978, ISRAEL.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

David Chen

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR013109

1. INTRODUCTION

In Prof. Ershov's opening lecture at the last WCCE (Laussane 1981); computer programming was metaphored as "the second literacy" and presented as a fundamental component in everyones culture. He forsees that: "we have no choice but to include (teaching computer programming) as operational congitive values into the structure and content of public education" (Ershov 1981).

Agreeing with this argument this paper presents a way to cope with spreading computer programming among the entire population.

The numerous problems related to teaching computer programming to young children may be divided into three areas:

The cognitive-research area: Computer programming is a rather complex activity, requiring tehnical control of the hardware, language acquisition ability, ability to solve problems in an algorithmic or procedural manner, and capacity for understanding abstract mathematical concepts. The knowledge of the cognitive skills and thought processes required for performing this activity by children, is limited. We do not know at what age children reach maturity and acquire the various concepts of programming language, and there is very little information regarding the factors explaining the large variability in achievements in the acquisition of a programming language and the ability to apply it, by children.

The curriculum development area

The existing curricula is poor in quantity and limited in variety: There are still only few curricula for teaching programming to elementary and junior high school children. The range of alternative projects aimed at different ages level, using different content and methodology offered to schools to choose from, is not sufficient yet.

The large quantity of books and magazines published in the last few years, which reflects the existing high demand for popular computing, is actually a "Technological-leisure time" industry that doesn't satisfy the need for appropriate school curriculum.

The School Implementation Area

The lack of curricula, rationale for its implementation, and adequate teaching materials and tools leaves teachers empty-handed while coping with the everyday classroom problems of computer programming teaching.

A rational for the curriculum development for teaching a programming language for children will be presented in this paper. This rationale was developed at the "Computers in Education Research Lab" in Tel Aviv University's School of Education. A Theoretical Descriptive Model of required skills needed by the naive programmer will be presented as a basis for the curriculum development and as a tool for localizing, mapping and understanding learners difficulties while acquiring a computer language. The guidelines and principles of the curriculum, its scope and sequence and teaching strategies will be listed. A description of the implementation and preliminary impressions and discussion will close this presentation.

2. RATIONALE FOR THE DEVELOPMENT OF A CURRICULUM FOR TEACHING PROGRAMMING LANGUAGE TO CHILDREN.

2.1 Descriptive Model of required Skills needed by the Naive Programmer

The early seventies Pioneer works set the fundamentals of a new interdisciplinary research area: The psychology of computer programming (Weinberg 1971; Sackman 1970; Papert 1970; Minsky 1970). Through the last decade the number of workers in this field has incresed considerably and some theoretical models describing computer programming as a complex

cognitive skill were developed. These models offer a classificaion of the wide range of skills and knowledge required for acquiring a computer language, into several main content domains and its hierarhical organization.

In the issue of dividing subject matter into main domains Shneiderman differentiates between two major types of knowledge required for programming: semantic knowledge, which includes understanding the structure of a computer program and its internal logic, and sintactic knowledge comprising the syntax and vocabulary of computer language (Shneiderman, 1977).

Suggestions for the hierarchical organization of programming language contents and skills appear in Mayer's eight level division of the Basic language from machine level up to program level (Mayer 1979), as well as in Shneiderman's spiral approach for teaching programming. (Schneiderman 1980).

A characterization of the naive programmer's cognitive activity appears in Brook's model, according to him understanding the problem, finding a method to solve it and coding the solution into Computer language are the three major activities of programming (Brooks 1977). In a similar manner, Green views programming as a problem solving tool (Green 1980).

None of the described models may serve as operational instrument in collecting, classifying, mapping and organizing the required programing basic skills during the course of an empirical study. Therefore a need arises for a model, which will serve as a basis for the development of a curriculum for programming teaching, and for locating and understanding the difficulties encountered by the child in the process of learning a programming language. The following describes a Model that intends to

satisfy this need.

2.1.1 Hierarchical Five Content Domain Model

There are two dimensions at the basis of the Model: the content dimension, dividing the skills and concepts into five content domain; and the hierarchical dimension, which organizes the components of each content domain hierarchically.

The five content domain

A. The technical skill of controlling the machine (T): This includes a variety of skills related to the ability to use the hardware: keyboard, peripheral equipment, etc. These skills may seem trivial; however, deficient technical control of the machine may cause difficulties during the acquisition of the concepts of a programming language.

B. Mastery of the programming language (L): This domain includes skills similar to those required for the acquisition of a new language. Programming language, like any other language, has its own syntax, vocabulary (commands) and grammatical rules. The programming student must learn the exact manner in which each command is written, and what it does, so that he may use it later as a building block in the program as a whole.

C. Understanding the structure of a computer program (S): This series of skills is related to the understanding of the structure of a computer program, the order in which commands appear in the program, and ability to follow their execution.

D. Understanding the logical dynamics of the computer program (D): This centers around the ability to follow the changes of values within the variables of the program while it is running, the ability to manipulate them and to apply mathematical-logical rules to these variables.

E. Various knowledge required, according to the program's topic (K):

When writing a computer program, specific knowledge related to the discipline or topic at hand is necessary. For example, when learning graphic programming it is necessary to understand the carthesian axis system, when dealing in musical programming, one ought to understand basic concepts in music, and in some areas of problem solving mathematical knowledge is essential.

The hierarchical dimension of the model

The various skills and concepts may be arranged hierarchically within each domain. This hierarchy may serve to determine the order of teaching. Each new skill will be based on components of knowledge which have been learned and mastered earlier.

A. The technical skill domain (T): The first skill required here is basic use of the keyboard and the ability to associate the specific key and the character appearing on the screen. At next stages the need to use special keys and cursor movement keys and the ability to use peripheral devices (diskettes, tape, printer, etc.) may become important. At the most advanced stages there is need for adding, or changing ROM and RAM cards, up to the level of a single chip.

B. The mastery of programming language domain (L): An hierarchical order of learning commands may be determined for each programming language. For example, in BASIC the hierarchy may start with elementary execution statements printing and graphic statements, as well as program operating commands; continue with assigning values to variables, input, and loops; up to complex programming statements such as subroutines and mathematical functions.

C. Understanding the structure of computer program (S): The continuum here starts with understanding the order of statements in a

linear program: in which execution proceeds from one statement to another, to the end of the program. Then the concept of loop is introduced, which leads to finite or infinite execution of a part of the program; next should come the follow-up of the order of statements in a complex program, composed of several blocks which are repeated according to specified conditions; understanding nested loops and recursion up to internalizing the principles of the Top Down design.

D. Understanding the logical dynamics of the computer program (D): The hierarchical continuum begins with the ability to conceive the variable as a location for keeping changing values in memory, through the ability to change the values of the variable, usage of the variable as a parameter in programs with single parameter and multi-parameter statements; understanding logic of operation of assignment and condition statements; up to manipulation, follow-up and use of variables of different types. The ability to follow the variable values while the program is working becomes more difficult as the program becomes more complicated.

E. The pre-requisite knowledge domain (K): Here the hierarchy is dependent on the specific discipline which is related to the problem being solved.

2.1.2 The model as a tool for describing the process of acquiring a programming language, and as a basis for curricular development:

The division of the skills required by the programmer into five content domains and the determination of a hierarchy of programming concepts within each such domain as suggested in the model may serve two objectives: a content infrastructure for curriculum development on one hand and an operational instrument for describing the process of acquiring programming language by novices, and a tool for mapping and explaining the

difficulties encountered by him, on the other hand. A curriculum was developed, it's contents have been defined according to the hierarchical organization of the content domains and it's principles and methods of teaching have been determined in agreement with the model. The activities of the learners and the difficulties encountered by them were mapped according to the content domains throughout a research and evaluation which have been caried out all through the curricular development. Within the conceptual framework of the model, one may describe two different types of learning each concept: the first is advancing in all the content levels in parallel, with regard to the hierarchy of concepts; the other is the integration of two or more levels into one concept.

2.2 The Principles of the Teaching Unit for Programming for Children:

The teaching Unit is intended for Grades 6 to 9, and in many cases parts of it may be taught as early as Grade 4. It is based on Microsoft-Basic language, and is intended to be used in microcomputers with graphic and sound capability. (e.g. Apple II, Atari, Comodore 64, Spectrum, etc.).

Throughout the curricular development, special emphasis has been put on the following:

A. Hierarchical organization of the material:

The material to be learned will be presented to the student hierarchically in all five content domain. Teaching the concepts and skills will be based on the components of knowledge which have been acquired and mastered earlier. Skipping over some skills which comprise a prerequisite, or advancing with the learning before mastery has been achieved, may cause difficulties later. For example, teaching the variable concept before teaching other basic concepts (as frequently seen in existing teaching programs) causes difficulties for beginning programmers, and children in

partiular. Each new skill will be pesented in small steps, attempting to integrate it with the knowledge which already exists by practice and application of the subject studied.

B. Hands on experience:

Learning will be accomplished by active work with a microcomputer, from the very beginning of the teaching and for most of its duration.

C. Softening the "mathematical blow"

In most existing programs, all the examples, exercises and contents to be learned are taken from mathematics. This has remained from the days in which computer teaching was carried out by people who worked in the discipline of applied mathematics, and was intended mostly for mathematicians and those specializing in this discipline. This is the source of many difficulties for students who do not have mathematical background. The present program attempts to disconnect the teaching of programming as much as possible from the mathematical contents. The examples and excercises will not require mathematical knowledge which is not related directly to computer programming.

D. Variated application areas:

In each stage of learning, the student will be requested to apply what he has learned using graphic, verbal and sound output. The student will create drawings, animations, sound effects, plan games, and write practical programs, related to his own world.

2.3. Scope and Sequence

The proposed curriculum for teaching a programming language to children may be divided into three main parts:

A. Basic blocks without variables

Here students will write single-loop programs without variables, which

perform simple tasks.

B. Basic blocks with variables:

This will concretrate on the use of variables and learning the principles for composing branched programs.

C. Complex programming:

Combinations of the basic blocks into projects integrating the basic concepts studied in the two previous parts, and the acquisition of princioples of top-down design.

The following list presents the contents of the chapters of the teaching unit:

Part A

Ch. Ø: First acquaintance with the keyboard; learning to operate the computer and peripheral equipment.

Ch. 1: Basic graphic statements.

Ch. 2: Computer program and its manipulation.

Ch. 3: The random number and its use as a parameter for graphic statements.

Ch. 4: Infinite loops.

Ch. 5: High resolution graphic statements.

Ch. 6: Printing statements.

Ch. 7: The computer as a calculator.

Ch. 8: Sound generation (dependent on the existence of suitable statements in the programming language).

Part B

Ch. 9: Variables and their use.

Ch. 1Ø: Manipulating variables.

Ch. 11: Input statements and various input devices.

Ch. 12: Conditioning and branching statements.

Ch. 13: Finite loops.

Part C

Ch. 14: Principles of structured programming.

Ch. 15: Building complex programs in graphic mode.

Ch. 16: Building complex programs in text mode.

The various content domains of the model are expressed in the above chapters. Each concept or topic are related basically to one content domain at least; it is also related in a secondary manner to other content domains. Table 1 shows the detailed representation of the content domains in the various chapters. For example, Ch. 11 which deals with input statements, describes the syntax of the Input statement (the language control level - L); stopping the program until input is supplied from the keyboard (understanding the structure of a computer program - S); the role of the input command for assigning values to variables (the logical dynamics of the computer program - D); and the various input devices are introduced (technical skills of controlling the computer - T).

Insert Table 1 here

2.4 The Teaching Methods

The lesson's plan

The textbook presents the concepts and topics of each chapter, subdivided into 4 sections: The first explains the concept, it's syntax and a simple example of it's use; the second includes a large number of short computer programs demonstrating various applications and uses of the concept and the statement, together with explanations regarding their role in the program; the third includes tasks and excercises of increasing difficulty, related to the statement or concept; the fourth gives suggestions for some projects in which the use of the new concept,

integrated with previous knowledge, is required.

The student's levels of activity and application:

The learning of each concept is accomplished at five levels of activity and application:

Level 1 - practice: The student feeds the computer with examples and existing programs, runs them, practices the syntax aspects and notes the association between the concept or statement and the output.

Level 2 - changing an existing program: The student changes a statement or a parameter in an existing program, in order to achieve the required result or output.

Level 3 - writing a similar program: The student writes a program which is similar to the existing program, in a different medium. At this level the student moves from the ability to change a single command or parameter in a command, to changing an entire program.

Level 4 - writing an original program: The student writes an original program based on the idea, principle or mode of application presented in the existing program.

Level 5 - project: During the course the student carries out comprehensive projects which integrates several principles and concepts into one program.

For example, chapter 4 presenting the infinite loop includes an existing program which draws two eyes, with pupils moving from right to left (by drawing a point, erasing it and redrawing it, etc.). At the first level, the student runs the program, trying to understand the algorithm which is performed and the meaning of the infinite repeating of it. At the second level of applicaion he is requested to change the colour of the pupil, change it from a point into a line, slow the pupil's movement, or increase

the distance between the two moving points. At the third level of application he is asked to write a program in which a line or a form move downwards. The fourth level of application demands that the student write an original program based on the simple animation principle which has been presented, e.g. a traffic light changing colours, a clock whose hands move in four situations, a tennis ball moving from one racket to another, etc. When high resolution graphics have been learned, the movement will be integrated in a comprhensive graphics project, e.g. a figure doing excercises in several situations, or a program presenting a machine with moving parts.

Programming Environment

In a curriculum which centers around active interaction with the computer an appropriate program environment is of fundamental importance. The curriculum programming environment consists of the use of a microcomputer with graphic capacity, easy to operate and use, line interpreter, and a full screen editor which simplifies the debugging process. The materials provided for the student are a guide book, a demonstration diskette, personal worksheets, concrete models, and graphic planning sheets.

The learning framework:

The learning may be carried out in class, in groups, or individually. Special care should be taken to ensure enough computer time for each child, not exceeding the ratio of 2-3 students per computer.

3. THE CURRICULUM'S IMPLEMENTATION

3.1 The Teaching and Research Scope

First impressions will be presented based on the teaching and research design that took place in the years 1982-1984.

The following groups were taught:

a) 10 classes of 7th-8th graders; b) 10 classes of 9th-11th graders; c) 2 classes of 6th grade disadvantaged children; d) 1 class of 6th grade gifted children

Parallely, two separate Research designs were implemented:

a) 6 case studies of children acquiring a computer language in optimal laboratory conditions - three 4th grade and three 6th grade aged children. Each child learned individually for approximately 50 hours having unlimited computer access time, a personal instructor and the developed instructional materials. As a result of close personal observation, detailed documentation of each child's learning process was obtained. A description of the research and its results appeared in a separate report (Nachmias, Mioduser, Chen, 1984).

b) An experimental study in which the curriculum was implemented in 4 elementary school classes: Two 4th grade and two 6th grade classes, for a period of 12 lessons (24 hours) in a regular school environment. A battery of tests was developed for each particular subject as well as a personal questionnaire summarizing students achievements. A detailed description of the research and it's results appears in a separate report (Nachmias, Mioduser, Chen, 1985).

3.2 Preliminary Impressions

Although research of computer language acquisition by children is far from complete, the author's findings may point to preliminary conclusions which will be described from three aspects: Cognitive, Affective and Curricular.

The Cognitive aspect:

In the course of studies, significant differences were found between

the age groups in acquisition of the computer language. As for 4th graders, most of them acquired the programming concepts presented in the first part of the curriulum (basic programming without variables), but, almost all of them found it very difficult to understand the concept of variables which is the core of the second part. On the other hand, most of the 6th graders succeeded in the second part as well as the first.

The time needed to learn and apply concepts, was significantly different between the age groups. In the given amount of time, some of the students in the eldest group (grades 9-12) reached successfully the objectives of the third part of the curriculum, whilst only a few of the 6th graders reached it. It may be possible that an addition of teaching time would have enabled more of the 6th graders to reach the same achievement, but probably not the 4th graders.

The analysis of students achievments from the point of view of the model's content domains show that the children, without almost any exception mastered the technical and syntactical aspects very rapidly. This observation was especially surprising among 4th graders that have not studied English yet (in Israel, English is learned as a foreign language starting at 5th grade). These students assimilated the new symbol system of the programming language and succeeded using it properly even without understanding the literal meaning of the statements. As an example, one of the 4th graders, while typing in (after 30 hours of instruction) a graphic program, misspelled the PLOT statement and typed "PLOTT" instead. When asked by his tutor to read his program to locate the mistake, the child answered astonished "But where am I supposed to know English from?!"

Difficulties were observed while acquiring the concepts of computer program and infinite loop that belong to the 3rd content domain, among most

of the 4th graders and part of the 6th graders. As studies proceeded, most of the students in all ages succeeded to understand these concepts. The fourth domain's concepts were an unpassable obstacle for 4th graders, but passable, with difficulty for 6th graders.

The difficulties that arose in the course of the students work may suggest about the nature of the cognitive process taking place while acquiring basic programming concepts and skills. The sources of difficulty can be divided into two parts: First - those caused by the icrease of complexity and abstracion of concepts within a content domain. Second - the difficulty which arises from the integration of several content domains.

An example of the first type of difficulty is the transition from understanding an assignment of a value to a variable (x=5) to the ability to change systematially the value of a variable (x=x+1), in the logical dynamics of the computer program domain (D).

An example of the second type is the integration of random number generating and infinite loops. Students who understood how infinite loops work and applied this knowledge in their original programs, and also understood the concepts ofrandom number generating ound it difficult to integrate both concepts in one program.

The Affective Aspect:

Motivation, enthusiasm and pleasure characterized most of the students work, during all the studies. In response to the concluding questionnaires in the experimental groups, all the students without exception answered that they "enjoyed" or "enjoyed very much" the studies and would like to continue them, even in school.

The attitude of most of the students to the machine was positive and

natural and their work flow was uninhibited, unlike some adults.

In spite of the high variance in achievements and application levels of concepts learned, the motivation of the weaker students was not lower than the stronger students. Each student succeeded applying what he had learned to his own capability and therefore was usually satisfied.

The Curricular Aspect:

Time seems to be a dominant factor in the success of acquisition of prgramming concepts. The time allotted to each group, which varied between 24 to 40 hours, was not sufficient even for the best students, to fully apply and acquire concepts from the second and third parts of the curriculum.

The various levels of activity in practicing and applying the material at the computer, were central in the learning process and durated two thirds of the learning time. To that, the variety of prepared programs and the assignments which appeared in the curriculum, contributed greatly. Nevertheless, a shortage of practice time to enhance mastery of concepts learned in every stage, was felt.

The use of graphics and application areas that are close to a child's world made the pleasant interaction between the child and the computer through programming activity, possible. Graphics enabled concretization of abstract concepts, and helped young children, and children who are afraid of mathematics, to cope with mathematical contents.

4. Conclusions and Discussion

The hierarchical structure of the curriculum and the model of the student's working process, contributed in making the learning process more fluent and minimizing the difficulties which the student have to cope with.

Softening mathematics contents and the use of computer graphics in

various applcation domains turn learning into a rich and creative process and enable lowering of the learning starting age. Hands on experience was central through the entire process. An essential condition in implementing such a program is to ensure sufficient computer time to each student.

Sixty hours seems to be a minimal period required for teaching properly all the three core parts of the curriculum. Because of the significant variance observed within and among the groups regarding required learning time it is recommended to study this issue more particulary.

In order to implement successully the curriculum it is necessary to enrich the repertoire of demonstration programs and exercises adapting them to the wide range of student's ages and capacities. The model presented may serve as a sorting and classification tool.

Research findings correspond to the classical level categorization of Piaget's theory. The concrete operational level 4th graders do not succeed in acquiring the formal concepts of the second part of the progress, concepts that were understood and applied by part of the early formal operation level 6th graders. However, the computer as a tool that enables the concretization of formal concepts may contribute to the understanding of formal concepts by chidren in transition between stages. Concepts like random numbers, variables, and logical operations receive a different appliable meaning while programing, far beyond that which is receiving during mathematics lessons.

Because of the high variance between children's levels of achievement, one can not definitely determine the optimal age for teaching computer programming. The chidren's curiosity, which is not entirely fulfilled by the formal education system, the success of the 4th graders in acquiring

and mastering concepts of technical and syntactical domains, and the beginning of maturity of the 6th graders in acquiring more abstract concepts, should be taken under consideration, while determining the starting age.

The curriculum developers and the teachers should consider the obstacles and difficulties which the students of various ages will cope with, while developing and teaching computer programming. The model presented here may serve as an effective tool for their classification as well.

We hope that the model and the curriculum developed will contribute to to a more adequate teaching of programming to the entire population.

References

(1) Brooks, R. "Towards a Theory of the Cognitive Processes in Computer Programming", Int. J. Man-Machine Studies 9, (1977), 737-751.

(2) Ershov, A.P., "Programming, The Second Literacy", in Lewis and Tagg (eds.) Computers in Education, 1-8, 1981.

(3) Green, T.R.G., "Programming as a Cognitive Activity", in Smith, T.H. and Green, T.R.G. (eds.) Human-Interaction With Computers, 1980, 271-320.

(4) Mayer, R.E., "A Psychology of Learning BASIC", Communication of the ACM, 22, (Nov. 1979), 589-593.

(5) Minsky, M., "Form and Content in Computer Science", Communication of the ACM, 17, April 1970, 197-215.

(6) Nachmias, R.; Mioduser, D.; Chen, D. Programming is a child's-game (Hebrew), Masada, in press, Israel, 1984.

(7) Nachmias, R.; Miodusar, D.; Chen, D. "A close look of how children acquire computer language: Six case studies.", (Hebrew), The Computers in Education Research Lab. Research Report No. 5, Tel-Aviv University, Israel, 1984.

(8) Nachmias, R.; Miodusar, D.; Chen, D." Teaching programming at school: a 4th and 6th grade experience" (Hebrew). The Computers in Education Research Lab., Research Report, No. 6. Tel-Aviv University, Israel, 1985.

(9) Papert, S., "Teaching Children Thinking", LOGO Memor No. 2, MIT, 1971.

(10) Sackman, A. Man-Computer Problem Solving, Averbach, Princeton, N.J., 1970.

(11) Shneiderman, B., "Teaching Programming: A Spiral Approach to Syntax

and Semantics", Computers and Education, 1 (1977), 193-197.

(12) Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Winthrop, Cambridge, Mass., 1980.

(13) Weinberg, G.M., The Psychology of Computer Programming, Van Nostrand Reinhold, New-York, 1971.

Table 1: Representation of Content Domains in the various chapters

Part	ch.	Chapter topic	Content Domains	
			Main	Secondary
	Ø	Learning to operate the computer	T	
	1	Basic graphic statements	L	T
	2	The computer program	S	L
	3	The random function	D	L
A	4	Infinite loop	S	L,D,T
	5	High resolution graphics	L	
	6	Printing statements	L	
	7	Computation	L	D
	8	Sound	L	
	9	Variables	D	L
	1Ø	Manipulating Variables	D	S
B	11	Input statements	S,D,T	L
	12	Conditioning statements	D	L
	13	Finite loops	S,D	L
	14	Structured programming	S,D	
C	15	Complex graphics programs	S,D	
	16	Complex alphanumerics programs	S,D	

T - Technical skill domain
L - Mastery of programming language
S - Understanding the structure of computer program
D - Understanding the logical dynamics of the computer program
K - Pre-requisite knowledge domain